www.icgst.com

Computer Vision in Low-Power Electronic Beehive Monitoring: In Situ Vision-Based Bee Counting on Langstroth Hive Landing Pads

Vladimir A. Kulyukin, Sarbajit Mukherjee
Department of Computer Science, Utah State University, 4205 Old Main Hill, Logan, UT 84322, USA
[vladimir.kulyukin, sarbajit.mukherjee]@aggiemail.usu.edu,
http://www.cs.usu.edu/people/

Abstract

An in situ computer vision method is presented for omnidirectional bee counting on landing pads of Langstroth beehives used by many apiarists worldwide. Bee counts are computed from static images of beehive landing pads. The presented method automates landing pad localization and improves bee counts through elimination of landing pad skew. Two bee counting algorithms are presented for counting bees on localized landing pads. The first bee counting algorithm is based on the 1D Haar Wavelet Transform. The second algorithm is based on contour analysis. The relative performance of both algorithms is evaluated on a sample of 793 images obtained from two electronic beehive monitoring devices deployed in live Langstorth beehives in northern Utah.

Keywords: *Wavelets, Haar Wavelet Transform, Contour Analysis, Electronic Beehive Monitoring, Sustainable Computing*

Nomenclature: *EBM - Electronic Beehive Monitoring, CV - Computer Vision, EBMD - Electronic Beehive Monitoring Device, RPi - Raspberry Pi, HWT - Haar Wavelet Transform*

1 Introduction

Professional and amateur beekeepers use visual estimates of forager traffic levels to evaluate bee colonies' health. Higher levels of forager traffic may indicate onsets of swarms or colony robbing activities; lower levels of forager traffic may indicate mite infestations, failing queens, or chemical poisonings [1]. Continuous and rapid advances in electronic sensors have made it feasible to transform individual beehives and entire apiaries into ad hoc sensor networks that monitor bee colonies' health in situ [2]. Electronic beehive monitoring (EBM) can also help researchers and practitioners to collect critical data on colony behavior and phenology without invasive beehive inspections [3].

In this article, an in situ computer vision (CV) method is presented for omnidirectional bee counting on landing pads of Langstroth beehives [4] used by many apiarists all over the world. Figure (1) shows the flow chart of the vision-based bee counting method realized in the software system described in this article.

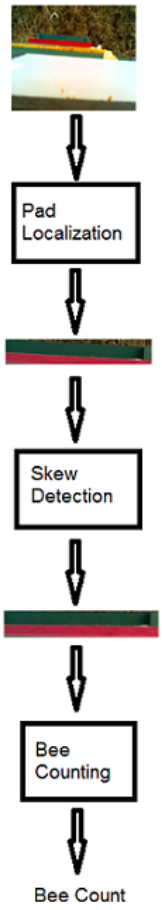

Figure 1: Flow chart of the method

As shown in figure (1), the system takes as input

a static image of a landing pad. The image is put through the pad localization module to localize the landing pad. Since the pad may be skewed, the image of the localized pad is processed by the skew detection module where the pad's skew angle is determined and the image is rotated accordingly. The localized and rotated pad is given to the bee counting module to compute a bee count, a non-negative real number approximating the number of bees on the pad.

The presented bee counting method is omnidirectional in that it does not distinguish between incoming and outgoing bee traffic: in situ bee counts are computed from static images of landing pads regardless of the actual orientations of the bees. Nor does the algorithm currently distinguish among three different types of bees: the worker, the drone, and the queen. The distinction of the three bee classes is beyond the current hardware capabilities of the presented EBM system in that it will likely require higher resolution cameras placed closer to landing pads. The identification of the queen bee on the landing pad is not a serious limitation, because, under normal circumstances, the mated queen never appears on the landing pad.

A fundamental objective of the reported research is to argue by demonstration that, given increasing availability and decreasing costs of CV hardware (e.g., miniature cameras and storage devices) and a proliferation of free, reliable open-source CV tools and libraries, CV is ready for inclusion in standard EBM sensor suites.

Another fundamental, long-term objective of this project is to create a suite of replicable hardware and software tools for citizen scientists to build their own EBM devices (EBMDs) and to promote the grassroots development of regional, national, and international cyberinfrastructures for capturing, sharing, and analyzing EBM data [5]. The hardware of the EBM system described in this article was assembled with off-the-shelf hardware components. The system's software was developed and tested with open source software tools.

The remainder of this article is organized as follows. In Section (2), related work is reviewed. In Section (3), the hardware and software details of BeePi [6], a multi-sensor, low-energy, solar-powered EBMD, are presented. Section (4) presents a method for localizing landing pads in images captured by BeePi. In Section (5), two bee counting algorithms are presented to count bees on localized landing pads. Section (6) describes an evaluation of both algorithms on a sample of 793 images and analyzes the results. In Section (7), conclusions are drawn.

2 Related Work

EBM is a mature research and development topic. In the 1950's, Woods, a sound engineer, designed and built Apidictor, an audio beehive monitoring tool [7]. Bencsik [8] equipped several hives with accelerometers and observed increasing amplitudes a few days before swarming, with a sharp change at the point of swarming. Evans [9] developed Arnia, a smartphone-based beehive monitoring system that uses weight, temperature, humidity, and sound. The system breaks down hive sounds into flight buzzing, fanning, and ventilating and sends digital alerts to beekeepers.

Ferrari et al. [10] assembled a system for monitoring swarm sounds in beehives. The system included a microphone, a temperature sensor, and a humidity sensor placed in a beehive and connected via underground cables to a computer in a nearby barn.

Rangel and Seeley [11] investigated signals of honeybee swarms. Five custom designed observation hives were sealed with glass covers. The captured video and audio data were monitored daily by human observers. The researchers found that approximately one hour before swarm exodus, the production of piping signals gradually increased and ultimately peaked at the start of the swarm departure.

In a longitudinal field investigation, Meikle and Holst [12] placed four beehives on precision electronic scales linked to data loggers to record weight for over sixteen months. The researchers investigated the effect of swarming on daily data and reported that empty beehives also had detectable daily weight changes due to moisture level changes in the wood.

Bromenshenk et al. [13] developed electronic SmartHive devices equipped with electronic scales, infrared bee counters, temperature and humidity sensors, digital weather stations, and wireless communication lines for Internet-based remote monitoring.

Silva et al. [14] proposed an intelligent trap with a laser sensor to selectively classify various insects, including honeybees, with audio signals. The researchers conducted an extensive evaluation of different feature sets from audio analysis and machine learning algorithms for the audio classification task. Combining multiple different classifiers based on the same feature set was shown to be ineffective while ensembles over different feature sets obtained from different signal representations performed better.

3 Hardware

The EBM literature, briefly reviewed in the previous section, indicates that CV remains relatively unexplored in EBM. However, our position is that CV is ready for inclusion in standard EBM sensor suites. CV is one of the virtual sensors of BeePi, a multisensor, solar-powered EBMD [5]. Each EBMD consists of a Raspberry Pi (RPi) computer, a camera, a solar panel, a temperature sensor, a battery, a hardware clock, a set of microphones, and a solar charge controller.

The exact hardware components of BeePi, shown in figure (2), include an RPi 3 Model B with 1GB


RAM, an RPi T-Cobbler, a full-size breadboard for sensor integration, a waterproof DS18B20 digital temperature sensor, an RPi Camera Board (CB), and a USB microphone hub where three microphones are plugged in. The wired microphones are connected to the RPi and placed into small holes drilled in hive walls to collect audio samples from inside the hive. The holes are covered with mesh nets on the inside to prevent propolisation of microphones by the bees.

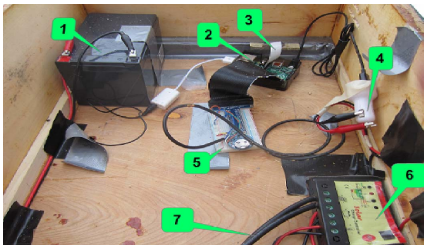

Figure 2: BeePi hardware components: 1) battery; 2) RPi; 3) RPi camera board; 4) car charger; 5) breadboard; 6) solar charge controller; 7) solar panel wires

Unlike the two previous hardware realizations of the BeePi EBMD described in [5, 6], the current hardware realization has the camera not only placed under a plastic cover but also attached to a wooden plank for improved balance and stability against wind. The plank is attached with screws and metallic brackets to the hive super where the BeePi hardware components reside, as shown in figure (3). The camera is further protected from the elements by a wooden box open at the bottom and attached to a migratory hive lid with screws and metallic brackets. When the lid is placed on the super with the BeePi hardware, as shown in figure (4), the box protects the camera against the elements from above and the four sides.

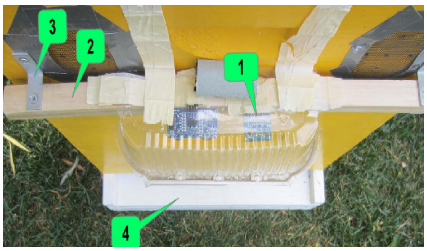

Figure 3: Camera (1) under plastic cover attached to wooden plank (2); plank is attached with metallic brackets (3) to box with BeePi hardware; camera looks down on landing pad (4)

Renogy 50 watts 12 Volts monocrystalline solar panels are used for harvesting solar energy into UPG 12V 12Ah F2 sealed lead acid AGM deep-cycle rechargeable batteries via Renogy 10 Amp PWM solar charge controllers. It takes approximately 25 minutes to wire a BeePi EBMD for deployment. Figure (5) shows the first author wiring two BeePi units for field deployment in May 2016.

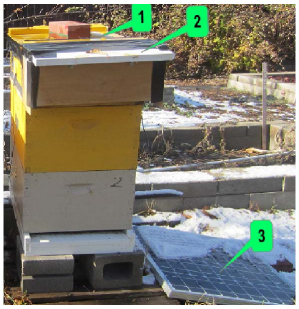

Figure 4: Hive lid (1) with protection box (2) on hive; solar panel (3) on ground

Data collection is done on the RPi with the software written in Python 2.7.9. The collected data are saved on a 32GB sdcard inserted into the RPi. When the system starts, three data collection threads are spawned. The first thread collects temperature readings every 5 minutes and saves them into a text file. The second thread collects 30-second wav recordings every 5 minutes. The third thread captures pictures of the hive's landing pad every 5 minutes and saves them as PNG. The bee counting method presented in this article works with these images. A cronjob monitors the threads and restarts them when necessary.

Figure 5: Wiring BeePi for field deployment

4 Landing Pad Localization

The proposed bee counting method consists of two stages: landing pad localization and bee counting on the localized bee pads. In the first stage, landing pads are localized and cropped from images. In the second stage, bees are counted on localized pads. This section describes the pad localization stage. Bee counting is discussed in Section 5.

A pad image, as seen in the left image in figure (6), is converted to grayscale and the average brightness is computed. If the brightness is below a threshold, which in the current implementation is set to 190, the original image is converted from `RGB` to `YCrCb`, where `Y` is the luminance component and `Cr` and `Cb` are the blue difference and red difference chroma components, respectively. The converted image is split into the `Y`,

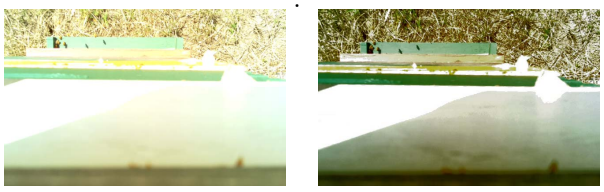

Figure 6: Left: original image; right: image with adjusted brightness

Cr, and Cb channels, and the histogram is equalized for each channel. The channels are merged into one image and the image is converted back to RGB, as shown in the right image in figure (6).

Figure 7: An image of a skewed pad

As shown in figure (7), some captured images have skewed landing pads. The skew may be caused by strong wind or by the physical manipulation of the beehive by a beekeeper. Therefore, the next step in the pad localization is to determine the skew angle of the pad and, if there is a skew, to rotate the image accordingly. Pad skew detection is based on the observation that the grass around the landing pad has more edges than the landing pad itself even when the bees are present on it. One morphological operation that captures this texture difference is corner detection, because the grass is naturally expected to have more corners than the landing pad.

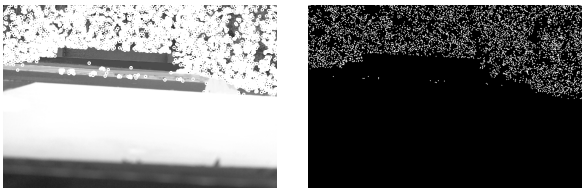

Figure 8: Left: corners in original image; right: true corner pixels

Toward that end, the image with adjusted brightness is processed with the Harris Corner detector [15] and the maximum pixel value is recorded as v_{max} (see the left image in figure (8)). A clone of the image with detected corners is eroded. The image with detected corners is compared with the image with eroded corners and only unmodified pixels are retained and thresholded at or above v_{max}. Then AND operation is applied to the image with detected corners and the image with thresholded corners, to delete all modified pixels. In figure (8), the left image shows the image with detected corners and the right image shows the image after the AND operation.

After the corners are detected, a hash map, $M_r(i) \rightarrow (c_0, c_1, ..., c_n)$, is created where i is a row number and $c_j, 0 \leq j \leq n$ are the columns of the detected corners in row r. Another hash map, $M_c(i) \rightarrow (r_0, r_1, ..., r_n)$, is created where i is a column number and $r_j, 0 \leq j \leq n$ are the rows of the detected corners in c.

The maps M_r and M_c are used to construct vertical and horizontal frequency projections, i.e., two corner frequency histograms H_r and H_c, respectively. In H_r, the bin numbers correspond to rows and the values of the bins are the counts of detected corners in those rows; in H_c, the bin numbers correspond to columns and the values of the bins are the counts of true corners in those columns. The images in figures (9) and (11) show H_r and H_c, respectively, computed for the image in figure (7).

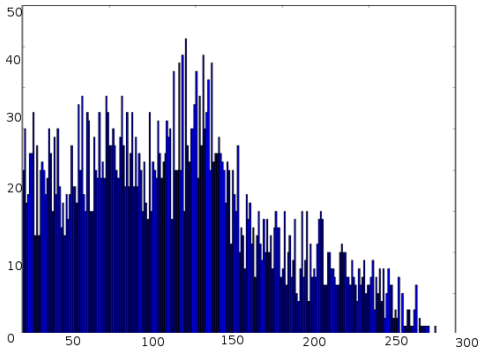

Figure 9: Row histogram H_r of image in figure (7)

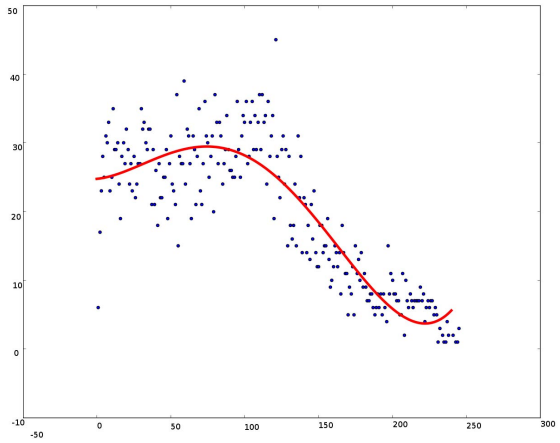

Figure 10: Curve fit $r(x)$ over H_r in figure (9)

The histograms H_r and H_c are used to detect declines in detected corner counts. Since the landing pads have fewer corners than grass, these declines indicate the upper corner points of the landing pad. To-

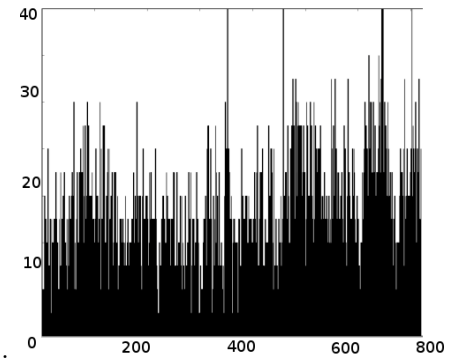

Figure 11: Column histogram H_c of image in figure (7)

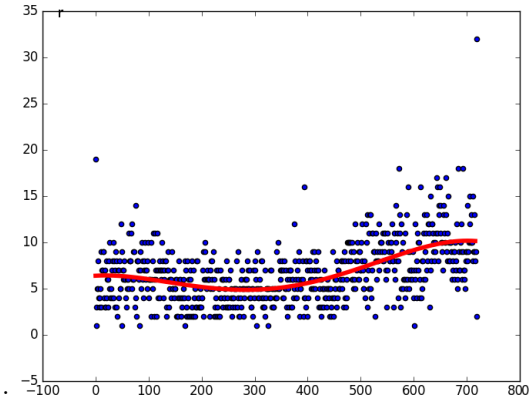

Figure 12: Curve fit $c(x)$ over H_c in figure (11)

ward that end, the histograms are converted into 2D scatter plots and the outliers in the plots are removed with the $L1$ distance in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates. A curve fitting procedure is applied to the corner counts to model the distribution of corner pixels with a 4-th degree polynomial. Equation (1) gives the curve for rows; equation (2) gives the curve for columns. The coefficients of x^4 and x^3 are very small in both curves, and are discarded.

$$r(x) = 0.0025x^2 + 0.017x + 24.7 \quad (1)$$

$$c(x) = -0.0001x^2 + 0.004x + 6.4 \quad (2)$$

The curves in equations (1) and (2) are used to locate the landing pad in a given image. Row-based localization uses $r(x)$ in equation (1) shown in figure (10). A window W_i^r of length 10 is created and shifted by 5 units left to right along the values of $r(x)$. Let M_i be the median of W_i^r and let $\Delta_i^r = M_i - M_{i-1}$. If $\Delta_i^r < 0$, the curve's slope is decreasing, which indicates the start of the landing pad. Let $I_r = \{i|\Delta_i^r < 0\}$ and let Q_{I_r} be the first quartile point of I_r. The first quartile point approximates the upper ordinate of the landing pad. Since the pad is approximately fifty pixels wide, given the distance of the camera from the landing pad, the lower ordinate of the landing pad in the image is computed as $Q_{I_r}+50$. Figure (13) shows the row-based cropping of the landing pad region obtained with this procedure applied to the image in figure (7).

Figure 13: Row-based cropping of landing pad region in image in figure (7)

After row-based pad localization, a similar procedure is executed for column-based pad localization using $c(x)$ in equation (2) shown in figure (12). From the curve one can see that there is a decline followed by a rise. A decline indicates a steady drop in the number of column corner counts, which, in turn, indicates the horizontal start of the landing pad. On the other hand, a rise indicates the horizontal end of the landing pad. A window W_j^c of length 10 is created and shifted by 5 units left to right along the values of $c(x)$. Let M_i^c be the median of W_j^c. Let $\Delta_i^c = M_i^c - M_{i-1}^c$. If $\Delta_i^c < 0$, the curve's slope is decreasing. Let $I_c = \{i|\Delta_i^c < 0\}$. Let M_s^c be the median of I_c. This value represents the left column where the landing pad is likely to start, i.e., the left horizontal start of the landing pad. Let $J_c = \{j|\Delta_j^c > 0\}$ and let M_e^c be the median of J_c. Then M_e^c is the right column index of the landing pad, i.e., the horizontal end of the landing pad. Figure (14) shows the column-based cropping applied to the image in figure (13).

Figure 14: Column-based cropping of landing pad region in image in figure (13)

As figure (14) indicates, there may be some grass left in the image after column-based cropping. To eliminate these grass spots, another iteration of column-based localization described above is executed after column-based cropping. Specifically, the corner detection algorithm is applied to the cropped image, as shown in figure (15), and the curve fitting procedure is applied to the corner counts to model the distribution of corner pixels with a 4-th degree polynomial, as shown in figure (16).

Figure 15: Detected corners in image in figure (14)

The left and right segments of the polynomial curve are ignored after the leftmost and rightmost non-zero frequency counts. For example, in figure 16,

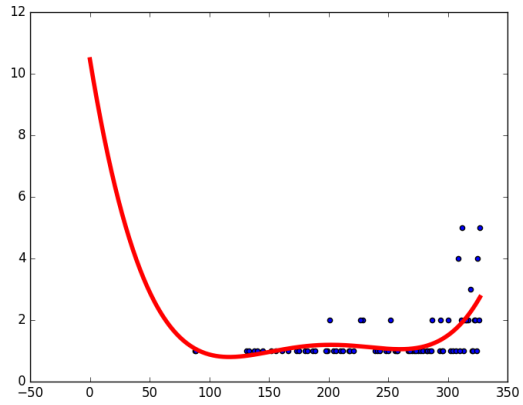

Figure 16: Fourth degree polynomial fit over corner distribution after 2^{nd} iteration of column-based pad localization applied to image in figure 15

the left falling segment can be safely ignored from 0 to the last non-zero column frequency count. The polynomial rise on the right, on the other hand, indicates the presence of grass. To detect the exact end of the landing pad, a window of 10 points W_i is created and shifted left to right 5 units at a time along the curve values. Let M_i be the median of the window points and let $\Delta_l = \{i|M_i - M_{i-1} < \theta\}, \theta = 0.105$ and $\Delta_r = \{i|M_i - M_{i-1} > \theta\}, \theta = 0.105$. If m is the middle of the image, two indices k and j are selected such that $k = min\{i|i \in \Delta_l \wedge i < m\}$ and $j = max\{i|i \in \Delta_r \wedge i > m\}$. The middle columns of W_k and W_j mark the left and right horizontal boundaries of the landing pad, respectively.

Figure 17: Reduced landing pad after 2^{nd} iteration of column-based plan localization applied to image in figure (14)

Figure 18: Corners detected in right half of cropped image in figure (14)

The final steps in landing pad localization are skew angle detection and image rotation. Pad skew detection is similar to the method we previously developed to detect text skew in nutrition labels on grocery products [16]. The input to the skew angle detection module is a horizontally and vertically cropped landing pad image like the one in Figure (17). The image is divided into the left and right halves. Another round of corner detection is executed on both halves. If the number of detected corners in either half is below a threshold the image is discarded. For example, the left half of the image in figure (14) does not have enough corners. However, the right half of the image in figure (14) does have some corners, as shown in figure (18).

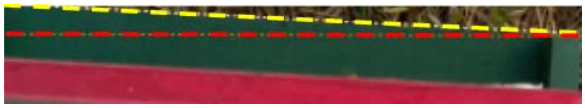

Figure 19: Skew angle detection in image in figure (15)

A list of rows, R_i, is computed in which there are no corners detected. Let $min(R_i)$ be the minimum row with no corners detected with row 0 being the top row in the image. Let $max(R_i)$ be the maximum row with no corners detected with row 0 being the bottom row in the image i.e scanning the image from bottom to top. If $|min(R_i) - max(R_i)| < \theta$, $\theta = 3$, the pad has no skew and does not need to be rotated. Otherwise, a column-based scan of the corner counts left to right indicates whether the corner counts increase left to right or decrease. A line is drawn between the points $(min(R_i), 0)$ and $(max(R_i), w-1)$, where w is the width of the image. This virtual line is depicted by a yellow dash line in figure (19). If the corner counts increase, the skew angle is $\angle(0,0)$, $(max(R_i), w-1)$, $(max(R_i), 0)$. If the counts decrease, the skew angle is $\angle(0, w-1), (max(R_i), 0), (max(R_i), w-1)$. In figure (19), the hypotenuse $[(0,0), (max(R_i), w-1)]$ of $\angle(0,0)$, $(max(R_i), w-1), (max(R_i), 0)$, is depicted with a red dashed line. Figure 20 shows the rotated image.

Figure 20: Pad image in figure (17) rotated counterclockwise to eliminate skew

5 Bee Counting

After the landing pad is cropped from the image, the bees are counted on it. This section presents two bee counting algorithms we have developed so far. In Section 6, the performance of these algorithms is evaluated on a sample of 793 images, which corresponds to the daytime periods of three consecutive calendar days.

5.1 Algorithm 1

The first bee counting algorithm is based on the 1D Haar Wavelet Transform (1D HWT) [17]. In the 1D HWT, a signal is a vector in $R^n, n = 2^k, k \in N$. Following the formalization in [18], let $W_a^{(k)}$ be a $2^k \times 2^k$

for computing k scales of the 1D HWT. This matrix can be effectively computed from the n canonical base vectors of R^n. If $x = (x_0, ..., x_{2^k-1})$ is a signal in R^n, then y is the k^{th}-scale 1D HWT of x is defined in equation (3).

$$W_a^{(k)} \cdot x^T = y \tag{3}$$

The values of the column vector y are defined in (4).

$$y^T = (a_0^{(0)}, c_0^{(0)}, c_0^{(1)}, c_1^{(1)}, ..., c_{2^{k-1}-1}^{(k-1)}) \tag{4}$$

In equation (4), $a_0^{(0)} = \mu(y)$ and c_i^j is the coefficient of the i^{th} Haar wavelet at scale j.

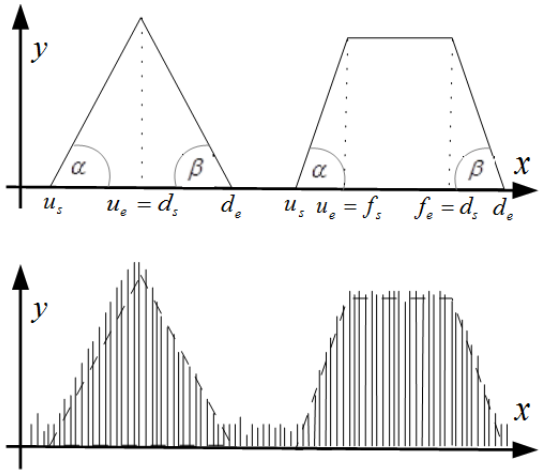

Figure 21: Up-down spikes

HWTs are used to detect significant changes in signal values [19]. We have previously proposed to classify signal changes as wavelet spikes [20]. Specifically, we propose four types of spikes to detect and classify signal changes: up-down triangle, up-down trapezoid, down-up triangle, and down-up trapezoid. Figure (21) shows up-down triangle and trapezoid spikes; figure (22) shows down-up triangle and down-up trapezoid spikes. In both figures, the lower graphs represent possible values of the corresponding Haar wavelets at a given scale k. The difference between up-down and down-up spikes is the relative positions of the climb and decline segments. In trapezoid spikes, flat segments are always in between the climb and decline segments. Formally, a spike S is a 9-tuple whose elements are real numbers given in (5).

$$S = (u_s, u_e, \alpha, f_s, f_e, \gamma, d_s, d_e, \beta) \tag{5}$$

The first two elements of the 9-tuple, u_s and u_e, are the abscissae of the beginning and end of the spike's climb segment, respectively, when the wavelet coefficients of the 1D HWT increase. This happens only if the samples taken from the analyzed signals also increase. If $cu_s^{(k)}$ and $cu_e^{(k)}$ are the k^{th} scale wavelet coefficient ordinates at u_s and u_e, respectively, then the sharpness of the spike's climb is measured as

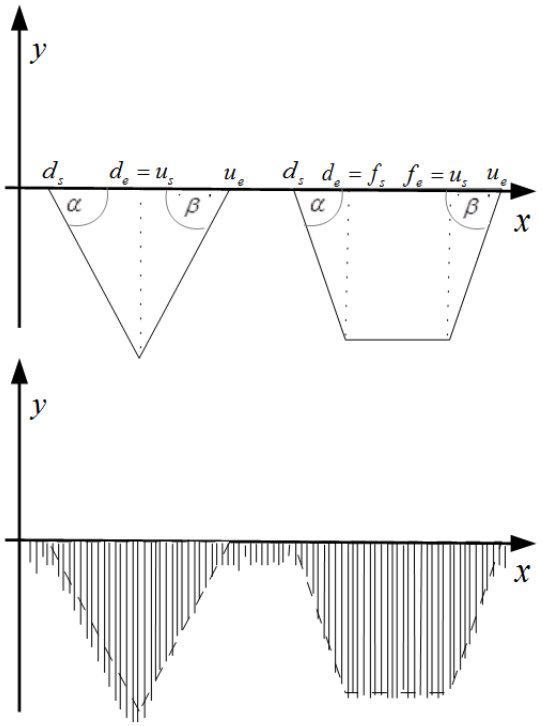

Figure 22: Down-up spikes

$\alpha = tan^{-1}(u_e - u_s + 1, cu_e^{(k)} - cu_s^{(k)})$, which is the third element of the 9-tuple.

The decline segment of the spike S in equation (5) is characterized by the seventh, eighth, and ninth elements of the 9-tuple: d_s, d_e, and β, where d_s and d_e are the abscissae of the beginning and end of the spike's decline segment, respectively, when the wavelet coefficients decrease. The decline occurs when the signal's samples decrease. If $cd_s^{(k)}$ and $cd_e^{(k)}$ are the k^{th} scale wavelet coefficient ordinates at d_s and d_e, respectively, then the sharpness of the spike's decline is measured as $\beta = tan^{-1}(d_e - d_s + 1, cd_e^{(k)} - cd_s^{(k)})$.

In trapezoid up-down or down-up spikes, the flat segment of the spike in equation (5) is characterized by the fourth, fifth, and sixth elements of the 9-tuple: f_s, f_e, and γ, where f_s and f_e are the abscissae of the beginning and end of the spike's flat segment, respectively, over which the wavelet coefficients either remain at the same ordinate or have minor ordinate fluctuations. If $cf_s^{(k)}$ and $cd_e^{(k)}$ are the k^{th} scale wavelet coefficients corresponding to f_s and f_e, respectively, the spike's flatness is estimated as $\gamma = tan^{-1}(f_e - f_s + 1, cf_e^{(k)} - cf_s^{(k)})$. The absolute value of γ is close to 0.

Given an image, spikes are computed for each row. Of course, depending on the application domain, column spikes can also be computed. When spikes are computed for row r, the column indices of the actual pixels covered by each spike at scale j are computed by the formula in equation (6), where s and e are the positions of the starting and ending wavelet

coefficients in the 1D HWT at scale j, respectively. For up-down spikes, $s = u_s$ and $d = d_e$, whereas, for down-up spikes, $s = d_s$ and $e = u_e$.

$$p(j, s, e) = \{i | 2^j \cdot s \leq i \leq 2^j \cdot (e + 1) - 1\} \quad (6)$$

Let n be the number of rows in an image and let U_r be the set of up-down spikes in row r, where $0 \leq r \leq n - 1$. The set of pixel columns in row r covered by the up-down spikes in r is given in equation (7), where j is a scale and s_z and e_z are the beginning and end positions of an up-down spike Z in row r, respectively.

$$Z^r_{U,j} = \bigcup_{z \in U_r} p(j, s_z, e_z) \quad (7)$$

Let D_r be the set of down-up spikes in row $r, 0 \leq r \leq n - 1$. The set of pixel columns in r covered by the down-up spikes is given in (8), where j is a scale and s_z and e_z are the beginning and end positions of a down-up spike Z in row r, respectively.

$$Z^r_{D,j} = \bigcup_{z \in D_r} p(j, s_z, e_z) \quad (8)$$

The number of unique column pixels covered by the up-down and down-up spikes in row r is given in equation (9). The formula in (10) gives the actual number of pixels covered by the up-down and down-up spikes in an image I with n rows.

$$Z^r_j = (Z^r_{U,j} - Z^r_{D,j}) \cup (Z^r_{D,j} - Z^r_{U,j}) \cup (Z^r_{D,j} \cap Z^r_{U,j}) \quad (9)$$

$$X_j(I) = \sum_{r=0}^{n-1} |Z^r_j| \quad (10)$$

For example, consider three 16 x 16 images in figure (23). Suppose it is required to separate bee pixels from non-bee pixels in the original bee image on the left. The bee pixels are those covered by up-down and down-up spikes in each row. The middle image in figure (23) shows the only up-down spike detected in row 8 after a single scale of the 1D HWT. The right image in figure (23) shows the only down-up spike detected in row 8. Both spikes are triangle ones.

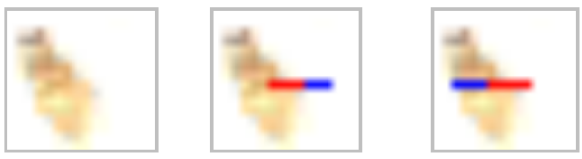

Figure 23: Bee image (left); up-down triangle spike in row 8 (middle); down-up triangle spike in row 8 (right); up-segments are red; down-segments are blue

The up-down spike, S_{ud}, shown in the middle image of figure (23), is defined in (11). Per notation in equation (5), the spike's climb segment starts at $u_s = 3$ and ends at $u_e = 4$. The angle of the climb is $\alpha \approx \pi/4$. Since this is a triangle spike, the flat segment is characterized as $f_s = f_e = -1$ and $\gamma = 0$. The spike's decline segment starts at $d_s = 5$ and ends at $d_e = 5$. The decline's angle is $\beta \approx \pi/3$.

$$S_{ud} = (3, 4, \pi/4, -1, -1, 0, 5, 5, \pi/3) \quad (11)$$

The down-up spike, S_{du}, shown in the right image of figure (23), is defined in (12). The spike's climb segment starts at $u_s = 3$ and ends at $u_e = 4$. The angle of the climb is $\alpha \approx \pi/4$. Since this is also a triangle spike, the flat segment is characterized as $f_s = f_e = -1$ and $\gamma = 0$. The spike's decline segment starts at $d_s = 1$ and ends at $d_e = 2$. The decline's angle is $\beta \approx \pi/3$. Note that since this is a down-up spike, the abscissae of the start and end of the decline segments are to the left of the abscissae of the start and end of the climb segment.

$$S_{du} = (3, 4, \pi/4, -1, -1, 0, 1, 2, \pi/3) \quad (12)$$

Using equations (6), (7), and (8), the pixel columns of the up-down and down-up spikes in (11) and (12), respectively, are given in equations (13) and (14), respectively.

$$Z^8_{U,1} = \{p(1, 3, 5)\} = \{i | 6 \leq i \leq 11\} \quad (13)$$

$$Z^8_{D,1} = \{p(1, 1, 4)\} = \{i | 2 \leq i \leq 9\} \quad (14)$$

Using equation (9), the set of pixel columns covered by the two spikes in row 8 in the left image of figure (23) is given in equation (15).

$$Z^8_1 = (Z^8_{U,1} - Z^8_{D,1}) \cup (Z^8_{D,1} - Z^8_{U,1}) \cup (Z^8_{D,1} \cap Z^8_{U,1}) \quad (15)$$

From (13), (14), and (15) one can compute the actual bee pixels in row 8, i.e., $\{10, 11\} \cup \{2, 3, 4, 5\} \cup \{6, 7, 8, 9\} = \{i | 2 \leq i \leq 11\}$. If the number of scales is $j = 1$, the number of bee pixels in the left image of figure (23) is given in equation (16).

$$X_1(I) = \sum_{r=0}^{15} |Z^r_1| = 44 \quad (16)$$

The pseudocode of the bee counting algorithm is given in figure (24). The algorithm takes an image I such as the image in figure (7), the normalizer N, and the number of scales j of the 1D HWT. The algorithm also takes the thresholds for the angles of up-down, flat, and down-up spikes, i.e., α, γ, β. In the current implementation, $\alpha = \beta = 60°$ and $= 5°$.

5.2 Algorithm 2

Algorithm 2 starts by image brightness adjustment and continues with the iterative reduction of the image to the actual landing pad to compensate for potential inaccuracies in landing pad localization outlined

```
1. procedure countBees1(I, N, j, α, γ, β)
2.     L = localizeLandingPad(I);
3.     gaussianBlur(L);
4.     pyramidMeanShiftFilter(L);
5.     maxRGBFilter(L);
6.     bleachBluePixels(L);
7.     convertToGrayscale(L);
8.     return ⌊X_j/N⌋;
```

Figure 24: Bee counting algorithm 1

in Section 4. As in algorithm 1 described in Section 5.1, bee counts are computed by dividing the total number of bee pixels by the average number of pixels occupied by individual bees obtained from camera calibration experiments.

Image brightness is maximal when the sun is directly above the beehive. When the sun is obscured by clouds, captured images become darker. To minimize the adverse effect of too much brightness or darkness, image brightness is dynamically adjusted to lie in in the range (45, 95), i.e., the brightness index should be greater than 45 but less than 95. This range was experimentally found to yield optimal results.

Figure 25: Adjusting brightness in image with localized landing pad

Figure (25) illustrates how brightness adjustment improves bee counts. The upper image on the right in figure (25) shows a green landing pad localized in the left image without adjusted brightness. The lower image on the right in in figure (25) shows a green pad identified in the same image with adjusted brightness. Only four bees were identified in the upper image on the right whereas in the lower image eight bees were identified, which is closer to the twelve bees found in the original image by a human evaluator.

$$H = \begin{cases} 60^\circ \left(\frac{G'-B'}{\Delta} \bmod 6 \right) & \text{if } C_{max} = R' \\ 60^\circ \left(\frac{B'-R'}{\Delta} + 2 \right) & \text{if } C_{max} = G' \\ 60^\circ \left(\frac{R'-G'}{\Delta} + 4 \right) & \text{if } C_{max} = B' \end{cases} \quad (17)$$

$$S = \begin{cases} 0 & \text{if } C_{max} = 0 \\ \frac{\Delta}{C_{max}} & \text{if } C_{max} \neq G' \end{cases} \quad (18)$$

To compensate for inaccuracies in landing pad localization described in Section 4, which may leave some small grass area above or on either side of the actual landing pad, algorithm 2 reduces the grass area present in the image of a localized landing pad. The three steps of grass area reduction are shown in figure 26. The first step is to convert the input from `RGB` to `HSV` according to equations (17) and (18). In equation (17), R', G', and B' are the R, G, B values normalized by 255, $C_{max} = max\{R', G', B'\}$ and $\Delta = C_{max} - max\{R', G', B'\}$. The value of V is set to C_{max}.

In the actual implementation, the format conversion is accomplished with the `cvtColor()` method of OpenCV. The `inRange()` method of OpenCV is subsequently applied to identify the areas of green or white pixels, the two colors in which the landing pads of our beehives are painted. Noise is removed through a series of erosions and dilations. The white pixels in the output image represent green or white color in the actual image and the black pixels represent any color other than green or white.

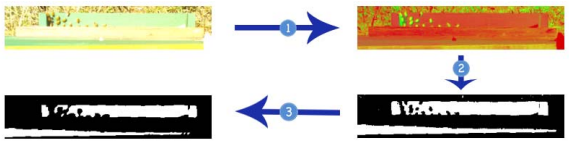

Figure 26: Landing pad identification steps: 1) HSV conversion; 2) color range identification; 3) noise removal

To further remove noise from the image and to reduce it as closely as possible to the actual landing pad, contours are computed with the `findContours()` method of OpenCV and a bounding rectangle is found for each contour. The bounding contour rectangles are sorted in increasing order by the Y coordinate, i.e., increasing rows. Figure (27) shows the bounding rectangles for the contours computed for the output image of step 3 in figure (26).

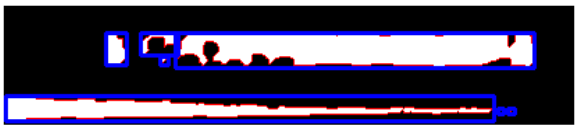

Figure 27: Bounding rectangles of found contours

Data analysis indicates that if the area of a contour is at least half the estimated area of the landing pad, the contour is likely to be part of the actual landing pad. On the other hand, if the area of a contour is less than 20 pixels, that contour is likely to be noise. In the current implementation of algorithm 2, the estimated area of the green landing pad is set to 9000 pixels and the estimated area of the white landing pad is set to 12000 pixels. These parameters can be adjusted for distance, depending on how far the BeePi camera is from the landing pad. Using this pixel area size filter, the approximate location of the landing pad is computed by scanning through all the contours in the sorted list and finding the area of each contour.

If the area is at least half the estimated size of the landing pad of the appropriate color, the Y coordinate of the contour rectangle is taken to be the mean Y coordinate and the scanning process terminates.

If the contour's area is between 20 and half the estimated landing pad area, the Y coordinate of the contour is saved. Otherwise, the current contour is skipped and the next contour is processed. When the first contour scan terminates, the mean Y coordinate, $\mu(Y)$, is calculated by dividing the sum of the saved Y coordinates by the number of the processed contour rectangles.

After $\mu(Y)$ is computed, the second scan of the sorted list of contour rectangles is performed to find all contours whose height lies in $[\mu(Y)-H, \mu(Y)+H]$, where H is half the estimated height of the landing pad for the appropriate color. In the current implementation, H is set to 20 for green pad images and to 25 for white pad images. The parameter H is hive dependent, because the exact camera location may differ slightly from hive to hive. For example, if the camera is placed closer to the landing pad, then H will have a higher value and if the camera is placed far from the landing pad, H will have a lower value.

Bounding rectangles are computed after the second scan to enclose all points in the found contours, as shown in figure (27). To verify whether the image has been sufficiently reduced to the actual landing pad, the area of the bounding rectangle is computed. If the area of the bounding rectangle is greater than the estimated area of the landing pad, the bounding rectangle may still contain noise, in which case another scan is iteratively performed to remove noise by decreasing H by 2 to 4 units. In most of the cases, this extra scan is not needed, because the landing pad is accurately found after the first two scans. Figure (28) illustrates the three steps of the contour analysis to reduce the image to the actual landing pad.

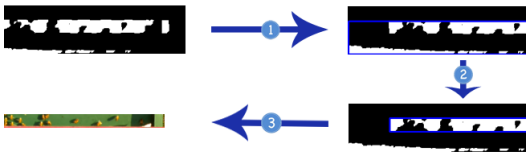

Figure 28: Contour analysis: 1) 1^{st} contour scan; 2) 2^{nd} contour scan; 3) pad cropping

After the pad is cropped, foreground and background pixels are separated on the basis of color. Specifically, in the current implementation of algorithm 2, for green landing pads, the background is assumed to be green and the foreground, i.e., the bees, is assumed to be yellow; for white landing pads, the background is assumed to be white and the foreground is assumed to be yellow. All pixels with shades of green or white are set to 255 and the remaining pixels are set to 0. Three rows of border pixels of the landing pad image are arbitrarily set to 255 to facilitate the upcoming bee counts. Figure (29) shows the output of background separation. In figure (29), the green background is converted to white and the foreground to black. Since noise may be introduced, the image is de-noised through a series of erosions and dilation with a 2×2 structuring element.

Figure 29: Background and foreground separation

To count bees in the image after background separation, the image is converted to grayscale and the contours are computed again. Data analysis suggests that the area of an individual bee or a group of bees vary from 20 to 3000 pixels. Therefore, if the area of a contour is less than 20 pixels or greater than 3000 pixels, the contour is removed. The area of one individual bee is between 35 and 100 pixels, depending on the distance of the camera from the landing pad. The green landing pad images were captured by a pi camera placed approximately 1.5 m above the landing pad with the average area of the bee being 40 pixels. On the other hand, the white landing pad images were captured by a pi camera placed approximately 70 cm above the landing pad where the average area of an individual bee is 100 pixels. To find the number of bees in green landing pad images, the number of the foreground pixels, i.e., the foreground area, is divided by 40 (i.e., the average bee pixel area on green landing pads), whereas, for the white landing pad images, the foreground area is divided by 100 (i.e., the average bee pixel area on white landing pads). The result is the count of bees in the image. In the upper image in figure (30), five bees are counted by the algorithm. The lower image in figure (30) shows the found bees in the original image.

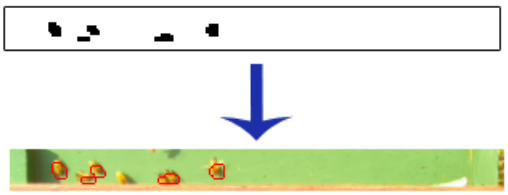

Figure 30: Omnidirectional bee counting

The pseudocode of algorithm 2 is given in figure (31). The first input parameter to this algorithm is the image I like the image in figure (20). The parameters C_l and C_u specify the lower and upper pixel areas for the contours. In the current implementation, $C_l = 20$ and $C_u = 3000$. The last parameter A_{bee} specifies the average pixel area of an individual bee. For green pads, $A_{bee} = 40$; for white pads, $A_{bee} = 100$.

```
1. procedure countBees2(I, C_l, C_u, A_bee)
2.   A = adjustBrightness(I);
3.   H = convertToHSV(A);
4.   P = cropLandingPad(H);
5.   B = separateBackground(P);
6.   CONS = identifyContours(B);
7.   AREA = 0;
8.   for each c in CONS
9.     if (area(c) > C_l and area(c) < C_u)
10.      AREA += area(c);
11.    end if
12.  end for
13.  BeeCount = ⌊AREA/A_bee⌋;
14.  return BeeCount;
```

Figure 31: Bee counting algorithm 2

6 Evaluation

To compare the accuracy of the two algorithms, 793 images of landing pads with bees on them were selected. Three human observers counted the bees on the pad in each image. The human counts served as ground truth. Both bee counting algorithms were evaluated on this sample of images. Table (1) summarizes the evaluation results. The first column gives the names of the two algorithms used. The numbers in the columns are error margin values that give the allowed margin of error between the human and computer counts for each image. For example, the value of the second column is equal to 3. Thus, the value 63.18 is the percent of images where bee counting algorithm 2 approached the human counts of the same images within a margin of 3, i.e., the absolute difference between the bees counted by the algorithm and the human count did not exceed 3. Similarly, the two values in the last column indicate that the first bee counting algorithm approached the human counts within a margin of 15 on 82.10% of the images, whereas the second bee counting algorithm approached the human counts within a margin of 15 on 90.29% of the images.

Algo/Error	3	5	7	10	15
Algo 1	39.97	53.72	63.56	73.77	82.10
Algo 2	63.18	73.77	80.10	84.99	90.29

Table 1: Evaluation of two bee counting algorithms: column names are error margin values; other values are percentages

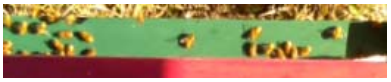

Figure 32: Inaccurately localized landing pad

The results in table (1) indicate that the second algorithm outperformed the first bee counting algorithm for each error margin. This relative underperformance can be attributed to the fact that, unlike algorithm 2, algorithm 1 does not compensate for inaccurate landing pad localization by reducing the size of the image to the actual landing pad. Figure (32) shows a typical image on which algorithm 2 outperforms algorithm 1. Note that the upper part of the image contains a sizable area of grass that algorithm 2 eliminates in steps 4 and 5 of the pseudocode in figure (31). Algorithm 1, on the other hand, detects quite a few spikes in the grassy area above the actual landing pad, which results in many false positives. When the landing pad is localized accurately both algorithms perform equally well. False positives were also caused by occasional shadows, leaves, or blades of grass wrongly counted as bees.

7 Conclusion

The investigation reported in this article continues our research on CV algorithms for omnidirectional bee counting [5, 6, 20]. The method for omnidirectional beecounting improves the previously proposed methods in three respects. First, the presented method completely automates landing pad localization whereas in our previously proposed methods the approximate landing pad coordinates were hardcoded into our program. Second, the presented method was implemented and evaluated in situ on a hardware device assembled with off-the-shelf hardware components. In our previous work, only image capture ran in situ whereas the implemented bee counting methods were evaluated in the laboratory on a standard desktop computer [5]. Finally, bee counts are improved through automatic detection and elimination of landing pad skews, which had a negative impact on bee counts in our previous experiments [6]. The obtained results indicate that CV can be included in standard EBM sensor suites. We hope that CV will help researchers, practitioners, and citizen scientists to build more reliable EBMDs and to promote the grassroots development of regional, national, and international cyberinfrastructures for capturing, sharing, and analyzing EBM data.

Acknowledgements

The first author, Vladimir Kulyukin, is grateful to Mr. Craig Huntzinger and Dr. Richard Mueller for letting him use their property in northern Utah for longitudinal EBM tests. The first author expresses his gratitude to Mr. Craig Huntzinger and Mr. Nathan Huntzinger for their help with inspecting monitored beehives and logging observations. All bee packages, bee hives, and beekeeping equipment used in this study were personally funded by the first author. The software system was implemented and tested exclu-

sively with open source tools.

References

[1] B. Walsh. A World without Bees. *Time*, August 19:26–31, 2013.

[2] M. T. Sanford. Second International Workshop on Hive and Bee Monitoring. *American Bee Journal*, December:1351–1353, 2014.

[3] M. E. A. McNeil. Electronic Bee Monitoring. *American Bee Journal*, August:875–879, 2015.

[4] Rev. L. L. Langstroth. *Langstroth on the Hive and the Honey Bee: A Bee Keeper's Manual*. Dodo Press, UK, 2008.

[5] V. A. Kulyukin, M. Putnam, and S. K. Reka. Digitizing Buzzing Signals into A440 Piano Note Sequences and Estimating Forage Traffic Levels from Images in Solar-Powered, Electronic Beehive Monitoring. In *Lecture Notes in Engineering and Computer Science: Proceedings of the International MultiConference of Engineers and Computer Scientists*, pages 82–87, Hong Kong, China, March 2016.

[6] V. A. Kulyukin and S. Reka. Toward Sustainable Electronic Beehive Monitoring: Algorithms for Omnidirectional Bee Counting from Images and Harmonic Analysis of Buzzing Signals. *Engineering Letters*, 24(3):317–327, 2016.

[7] E. F. Woods. *Means for Detecting and Indicating the Activities of Bees and Conditions in Beehives. Patent 2,806,082*. U.S. Patent Office, 1957.

[8] M. Bencsik, J. Bencsik, M. Baxter, A. Lucian, J. Romieu, and M. Millet. Identification of the Honey Bee Swarming Process by Analyzing the Time Course of Hive Vibrations. *Computers and Electronics in Agriculture*, 76:44–50, 2011.

[9] W. Blomstedt. Technology V: Understanding the Buzz with Arnia. *American Bee Journal*, October:1101–1104, 2014.

[10] S. Ferrari, M. Silvab, M. Guarinoa, and D. Berckmans. Monitoring of Swarming Sounds in Bee Hives for Early Detection of the Swarming Period. *Computers and Electronics in Agriculture*, 64:72–77, 2008.

[11] J. Rangel and T. D. Seeley. The Signals Initiating the Mass Exodus of a Honeybee Swarm from its Nest. *Animal Behavior*, 76:1943–1952, 2008.

[12] W. G. Meikle and N. Holst. Application of Continuous Monitoring of Honey Bee Colonies. *Apidologie*, 46:10–22, 2015.

[13] J. J. Bromenshenk, C. B. Henderson, R. A. Seccomb, P. M. Welch, S. E. Debnam, and D. R. Firth. Bees as Biosensors: Chemosensory Ability, Honey Bee Monitoring Systems, and Emergent Sensor Technologies Derived from the Pollinator Syndrome. *Biosensors*, 5:678–711, 2015.

[14] D. F. Silva, V. M. A. de Souza, G. E. A. P. A. Batista, D. P. W. Ellis, and E. Keogh. Applying Machine Learning and Audio Analysis Techniques to Insect Recognition in Intelligent Traps. In *Proceedings of the IEEE 12th International Conference on Machine Learning and Applications*, pages 99–104, Miami, Florida, USA, December 2013.

[15] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, Manchester, UK, 1 August–2 September 1988.

[16] T. Zaman, V. A. Kulyukin, and A. Cutler. Text Skew Detection in Printed Text Images Relying on 2D Haar Wavelets. *Journal of Graphics, Vision and Image Processing (GVIP)*, 16(1):41–50, 2016.

[17] A. Jensen and A. Cour-Harbo. *Ripples in Mathematics: the Discrete Wavelet Transform*. New York: Springer, 2001.

[18] Y. Nievergelt. *Wavelets Made Easy*. Boston: Birkhäser, 2001.

[19] S. Mallat and W. Hwang. Singularity detection and processing with wavelets. *IEEE Trans. on Information Theory*, 38(2):617–643, 1992.

[20] V. A. Kulyukin. In situ omnidirectional vision-based bee counting using 1d haar wavelet spikes. In Lecture Notes in Engineering and Computer Science: Proceedings of the International MultiConference of Engineers and Computer Scientists, pages 182–187, Hong Kong, China, March 15–17 2017.

Biographies

Vladimir A. Kulyukin is an Associate Professor of Computer Science at Utah State University. He holds a Ph.D. in Computer Science from the University of Chicago that he received in 1998. His research interests include computer vision, sensor fusion, wavelets, and sustainable low-power computing.

Sarbajit Mukherjee is a graduate student of Computer Science at Utah State University working under the supervision of Dr. Kulyukin. He holds an M.S. in Computer Science from the Indian Institute of Engineering Science and Technology in Shibpur. His research interests include computer vision and machine learning.